From Monologue to Dialogue: Natural Language Generation in OVIS*

Mariët Theune
Parlevink Language Engineering Group, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
theune@cs.utwente.nl

Abstract

This paper describes how a language generation system that was originally designed for monologue generation, has been adapted for use in the OVIS spoken dialogue system. To meet the requirement that in a dialogue, the system's utterances should make up a single, coherent dialogue turn, several modifications had to be made to the system. The paper also discusses the influence of dialogue context on information status, and its consequences for the generation of referring expressions and accentuation.

Introduction

Many practical dialogue systems use canned prompts for output generation. This may be sufficient in restricted domains where only a limited number of fixed system utterances is required, but for systems where a high number of varying utterances must be generated, a more advanced form of natural language generation is required. Ideally, the chosen generation method should be context-sensitive, for instance adapting the wording of the generated utterances to the user's word choice and to the current state of the dialogue. In addition, it should be able to generate not only short system prompts but also longer stretches of text (e.g., information presentations in response to a user's query). Finally, in the case of spoken dialogues, it is also desirable if the natural language generation component can produce prosodic information for use by speech synthesis.

A system that meets the above requirements is D2S, a concept-to-speech system that was originally developed for the generation of monologues (van Deemter & Odijk 1997, Theune *et al.* 2001). This paper discusses how D2S was adapted for the generation of system utterances in a spoken dialogue system called OVIS.

The paper starts with a general overview of the OVIS system, focusing on its dialogue management, language and speech generation components. The next section describes how the use of syntactic templates in the language generation component had to be adapted for the generation of dialogue utterances. Then, it is briefly discussed how a dialogue context may influence referring expression generation and accent assignment. Finally, some concluding remarks and future work are presented.

The OVIS system

OVIS[1] is a Dutch spoken dialogue system that provides information concerning train connections and ticket prices. It was developed as part of a Dutch national research project, the NWO Priority Programme *Language and Speech Technology*. The general architecture of the OVIS system is shown in Figure 1. Below, the system's dialogue manager and its interaction with the language generation module are described, and the language and speech generation components are introduced. More information on the other modules of OVIS can be found in Strik *et al.* (1997) (speech recognition), van Noord *et al.* (1999) (rule-based language analysis) and Bod (1998) (statistical language analysis).

Dialogue management in OVIS

The OVIS dialogue manager (Veldhuijzen van Zanten 1998) is responsible for coordinating the dialogue with the user and consulting the database of train connections and ticket prices. Based on input from the language analysis module, which interprets the results from speech recognition, the dialogue manager determines which actions must be carried out in the next system turn. It then sends a message to the language generation module (LGM), specifying the information or dialogue act that must be expressed. The messages refer to a data structure representing the information which can be talked about

*This research was carried out within the Priority Programme *Language and Speech Technology*, sponsored by NWO (Netherlands Organisation for Scientific Research).

[1] *Openbaar Vervoer Informatie Systeem* ("Public Transport Information System")

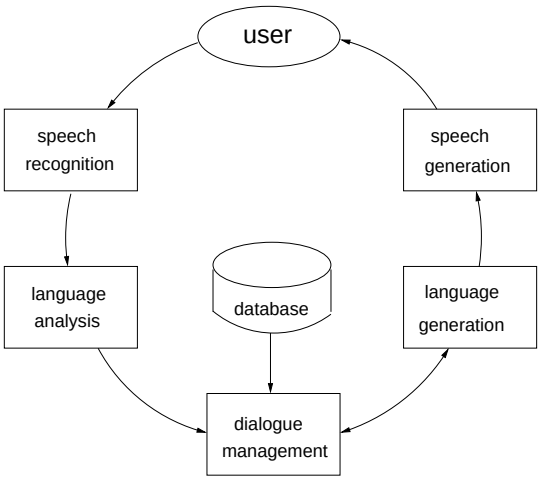

Figure 1: OVIS architecture.

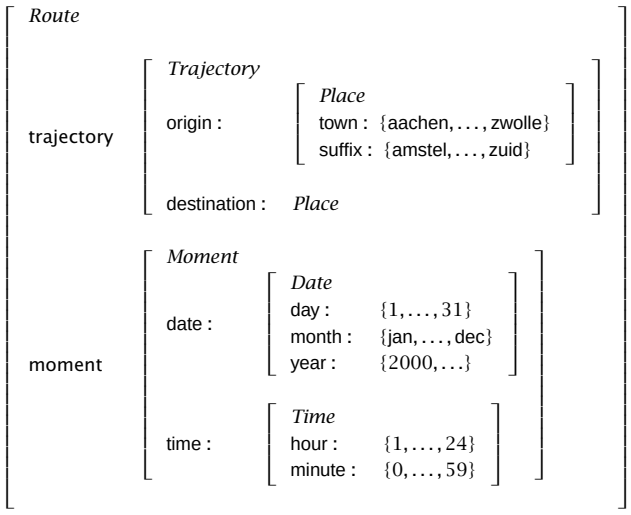

Figure 2: The route slot.

by the user and the system, in the form of a typed hierarchy of slots and values. Figure 2 shows the data structure's route slot, which holds the values needed to retrieve train time table information from the database. These values are filled in during the dialogue with the user.

The messages which the dialogue manager sends to the LGM for expression can be divided into two basic classes: dialogue prompts, which are part of the dialogue in which the system tries to determine the user's wishes, and presentations of information from the database. In this paper, only the generation of the dialogue prompts is discussed. There are two general types of dialogue prompts: *information requests*, where the system asks for the details of the user's intended journey, and *meta-acts*, utterances about the dialogue itself that may be used to deal with communication problems. The most important meta-act is *verification*: due to imperfect speech recognition, the system constantly has to verify whether it has understood the user's utterances correctly. Below, a few example messages are given,[2] together with the corresponding utterance generated by the LGM, and an informal description. All examples in this paper are given in translation.

(1)

Information requests:

DM: `the_route.trajectory.origin ??`
LGM: *From where do you want to leave?*
(Open question for the departure station)

DM: `the_price.class ?| {first | second }`
LGM: *Do you want to travel first or second class?*
(Alternatives question for the travel class)

DM: `the_return_trip.moment.date.day ?= 3`
LGM: *Do you want to return on the third?*
(Yes-no question for the day of return)

Meta-acts:

DM: `the_route.trajectory.origin ?+ asd`
LGM: *Departure from Amsterdam Central Station?*
(Verification of the departure station)

DM: `the_route.trajectory.origin.town **uden`
LGM: *Uden does not have a train station.*
(Noting an invalid departure town)

The messages shown in (1) are all 'simple' in the sense that they only involve one slot in the data structure. Complex messages are also possible, for instance combining two verifications:

(2)

```
DM: the_route.(moment.time ?+ 10:30;
                trajectory.destination ?+ mas)
```
LGM: *To Maarssen, at ten thirty?*
(Verification of destination and time of travel)

The dialogue manager employs a so-called *zooming* strategy: it begins with asking open questions about slots high up in the hierarchy, but when necessary, it zooms in towards more specific questions about lower-level slots. For example, at the start of the dialogue the user is asked what kind of information he or she wants; a very high-level open question

[2] The messages shown in this paper are slightly abbreviated: instead of specifying the full path through the OVIS data structure, they are shown starting from the level of the route slot.

which leaves the user a lot of room for initiative. An experienced user can react by stating all travel details in one turn, and get the dialogue over with quickly. A less experienced user, however, could be at a loss about how to respond. When this is detected, the dialogue manager sends a more specific message, e.g., asking whether the user is interested in route or price information. For more details on this adaptive, mixed-initiative dialogue management strategy, see Veldhuijzen van Zanten (1998).

Language and speech generation

The system utterances in OVIS are generated by D2S, a concept-to-speech framework that is to a large extent domain and language independent. D2S was originally developed for the DYD (Dial Your Disc) system, which generates English monologues about Mozart compositions (van Deemter & Odijk 1997). D2S also formed the basis for the GoalGetter system, which generates Dutch soccer reports (Theune *et al.* 2001). In D2S, language and speech generation form separate, reusable modules, which are connected through a prosodic component embedded in the language generation module (LGM). The prosodic component uses contextual and syntactic information provided by language generation to compute the placement of pitch accents and phrase boundaries in the generated utterances. This gives a better prosodic output quality than can be achieved when feeding plain text into a text-to-speech system.

In the LGM, sentences are generated from so-called *syntactic templates*, which combine TAG-like syntactic structures with conditions that determine if they are suitable given the current state of the context model. Each generated sentence leads to an update of this model, recording which information has been expressed, which words have been used, which discourse objects have been mentioned, etc. Examples of syntactic templates and their use in OVIS are given in the next section. In addition to checking the conditions on the syntactic templates, the context model is used for the generation of referring expressions. For instance, an anaphoric expression may be generated when an appropriate antecedent is found in the context model.

The prosody component of the LGM enriches each generated sentence with markers for pitch accents and phrase boundaries. It uses the context model to determine the information status of the concepts and entities being referred to: new, given or contrastive.[3] Words expressing new information are always accented, whereas words expressing given (previously mentioned) information are deaccented, except in the case of contrast (Theune 2002). The exact placement of pitch accents and phrase boundaries within a sentence is determined using the syntactic information from the templates. In-depth discussions of prosody computation in D2S are given in Theune *et al.* (1997) and Theune *et al.* (2001).

Finally, the speech generation module of D2S transforms the enriched text string it receives from the LGM into a speech signal. Two different speech generation methods are available, both of which have been used for speech generation in OVIS (in different versions of the system). The first method is diphone synthesis, which concatenates small speech segments consisting of the transition between two adjacent phonemes. The second method, which offers higher speech quality but less flexibility, is an advanced form of phrase concatenation. Here, speech is produced by concatenating phrases of varying length that have been recorded in different prosodic versions. Both available methods make use of the prosodic information provided by the LGM. More information on speech generation in OVIS can be found in Klabbers (2000).

The LGM from monologue to dialogue

In OVIS, some of the responsibilities the LGM has when it is used in 'monologue mode' are taken over by the dialogue manager. In OVIS, it is the dialogue manager which determines what to say at which point in the dialogue, leaving it to the LGM to determine how to say it. The practical advantage of this task division is the strict modularization of the system, with language-independent dialogue knowledge residing in the dialogue manager, and linguistic knowledge in the generation component. In addition, this set-up where the LGM has to carry out the instructions from the dialogue manager as they come, fits in well with the local, reactive planning approach of the LGM outlined above. Nevertheless, to deal with the new demands of language generation in a dialogue context, several adaptations had to be made to the LGM's template selection process. These are discussed below. First, some example OVIS templates are presented.

Syntactic templates in OVIS

Figure 3 shows three syntactic templates from OVIS that can be used for the generation of elliptic verification questions. They have been simplified by leaving out some information which is less relevant here (such as the full syntactic trees). The main elements of the syntactic templates are the following.

The first element, *S*, is a syntactic sentence tree with gaps in it for variable information. Many of the syntactic trees in the OVIS templates are minimal, in that only the head of the construction is lexicalized and the gaps coincide with its arguments. This holds in particular for verification questions, which are generally formulated as elliptic utterances (to keep the system prompts as shorts as possible). Those in Figure 3 only consist of a PP (e.g., *To Am-*

[3] The last notion is orthogonal to the first two, as both given and new information may be contrastive.

sterdam?), of which only the head is lexicalized in the syntactic tree.

The second element, E, is a list of calls to so-called Express functions that are used to fill the gaps in the syntactic tree. Compared to monologue generation, in OVIS the role of the Express functions is somewhat limited. The most frequently occurring variables correspond to towns and train stations, and these are usually referred to by their proper name, except for rare occasions where the anaphor *there* can be used. Other frequent expressions in OVIS are times and dates.

The third element, C, lists the local conditions on the syntactic template. These refer to the actions the LGM has to carry out in the current turn, and to the context model. For instance, the condition on Template DestVerif from Figure 3 says that this template is applicable when the value of the destination slot must be verified. The template's additional conditions on the context model are not shown here.[4] After the template has been applied, the LGM updates the context model to record that a verification question about the destination has been asked. It also records in which dialogue turn this was done.

The last element shown in Figure 3, L (*level*), was added specifically for the purpose of dialogue prompt generation. It specifies how many slots from the OVIS data structure are dealt with by the syntactic template. The reason for adding this element is explained in the next section.

Template DestVerif

S = To <**dest**>?
E = **dest** ← Express(*the_route.trajectory.destination*)
C = tobeverified (*the_route.trajectory.destination*)
L = 1

Template TimeVerif

S = At <**time**>?
E = **time** ← Express(*the_route.moment.time*)
C = tobeverified (*the_route.moment.time*)
L = 1

Template DestTimeVerif

S = To <**dest**>, at <**time**>?
E = **dest** ← Express(*the_route.trajectory.destination*)
time ← Express(*the_route.moment.time*)
C = tobeverified (*the_route.trajectory.destination*) ∧
tobeverified (*the_route.moment.time*)
L = 2

Figure 3: Example OVIS templates (simplified).

[4]For reasons explained below, this and the other templates from Figure 3 cannot be used when the system has attempted to verify the same value(s) in its previous turn.

Saying it all in one turn

When generating dialogue prompts, the LGM faces the following restrictions: (*i*) the message from the dialogue manager should be expressed within one turn, and (*ii*) a turn may contain only one question. The reason for the latter restriction is that a question is a 'release-turn' action, i.e., an action which signifies the end of a turn and indicates that the other dialogue participant may now take the floor (see Traum (1994):61). After a question has been generated, the user generally starts answering it immediately. Generating more than one question within a turn is therefore likely to lead to interruptions and confusion.

The above requirements are especially important when the LGM has to express a message concerning more than one slot from the data structure, such as (2), where the values of both destination and time must be verified. In principle, in this situation all three templates shown in Figure 3 are applicable. In 'monologue mode', the LGM would just pick one of these at random, and would go on selecting syntactic templates and generating sentences from them until the entire message has been expressed. If either Template DestVerif or Template TimeVerif were to be picked, this strategy would lead to the generation of two consecutive questions, as in (3).

(3)

LGM: *At ten thirty? To Maarssen?*

Since these two questions are prosodically separated by a final phrase boundary, the user might assume that the turn ends after the first question, and start answering too early. To keep this from happening, the template selection strategy has been changed so that the LGM always picks the syntactic template that deals with the highest possible number of slots, thus generating only one, complex question. To achieve this, the level component L has been added to the syntactic templates, and the generation algorithm has been adapted so that during generation, higher level templates are preferred over lower level ones. In example (2), this causes Template DestTimeVerif to be chosen for application, as it has a higher level (i.e., deals with more slots) than both other applicable templates. As a consequence, in (2) one question is generated that verifies the values of both destination and time, rather than two consecutive questions as in (3).

An alternative approach would be to use only one of the two lower-level templates, and stop after the corresponding question has been generated. However, this means that the instructions from the dialogue manager are not fully carried out. The LGM only resorts to this when there is no template available that can express the entire message in one turn. Such cases are discussed in the next section.

Turn coherence

In principle, a message specified by the dialogue manager can contain any combination of slots from the data structure, as in (2). However, some slot combinations are difficult to express in one coherent system turn. An example is (4), where the LGM is instructed to verify whether the user wants to have route information (as opposed to price information), and whether the user wishes to travel at ten thirty. These verifications cannot be combined in one question without the result being either long and awkward, as in LGM(a), or severely underspecified and thus incomprehensible, as in LGM(b).

(4)

DM: `( variant ?+ the_route;`
`the_route.moment.time ?+ 10:30)`
LGM (a): *Do you want to have route information, and do you want to travel at ten thirty?*
LGM (b): *Route information, at ten thirty?*

To avoid the generation of utterances like LGM(b) above, and to ensure the coherence of the system utterances, syntactic templates have been constructed for combinations of *related* slots only. Two slots are regarded as related if they share the same mother or grandmother slot in the OVIS data structure. Intuitively, these slots seem to provide the best combinations for complex verification questions,[5] but the choice for this specific relation has not been empirically validated. Examples of related slots are origin and destination (mother: trajectory) and time and destination (grandmother: route).

Since the slots involved in (4) are not considered to be related, there is no syntactic template available to verify the values of both slots at once. Therefore, to express the entire message two different syntactic templates must be used, each verifying the value of one of the specified slots. However, as was explained in the previous section, first using one template and then the other is not an option, as it will result in the asking of two questions in one turn. Therefore the generation algorithm was modified to apply the first template that happens to be selected,[6], and then stop upon detection that a question has been generated. As a result, only a part of the input message is expressed. To inform the dialogue manager of this, the LGM sends a feedback message indicating which part of the input message has actually been expressed. This is illustrated in (5), where the feedback from the LGM to the DM is given below the generated utterance.

[5]A practical side-effect of this filter is that it greatly reduces the number of templates to be constructed.

[6]Since both applicable templates have the same level, the LGM picks one of them at random.

(5)

DM: `( variant ?+ the_route;`
`the_route.moment.time ?+ 10:30)`
LGM: *At ten thirty?*
`the_route.moment.time ?+ 10:30`

Context-sensitive template selection

Most of the monologue generation systems in which D2S has been used can be characterized as 'infotainment' applications, which are aimed at getting information across *in a pleasant way*. In such applications, having variation in the generated output is important to keep users from getting bored when listening to several presentations in succession. One way of achieving this variation is to make a random choice between syntactic templates that express the same content but use different wording. Information systems like OVIS lack the entertainment aspect: their primary goal is to get information across to the user as clearly, quickly and efficiently as possible. In such systems gratuitous variation in the system output is unwanted, as users may try to interpret this kind of variation as meaningful even though it is not. In addition, the lack of predictability caused by variation may hamper the user's processing of the system utterances.

In OVIS therefore, there is no arbitrary choice between 'equivalent' syntactic templates. This does not mean, however, that each message is always expressed using the same template. Different templates are available that express the same message using different wording. Which of these is selected, depends on the dialogue context. This can be illustrated by (6), a dialogue fragment in which the dialogue manager attempts to verify the value of the destination slot twice in succession, due to a speech recognition error. The first time, the LGM expresses the verification question in the standard way, using an elliptic utterance. When the user's answer is not properly recognized, the dialogue manager repeats the question. This time, the LGM selects another template, expressing the verification question using a more elaborate, non-elliptic formulation.

(6)

DM: `the_route.trajectory.destination ?+ mas`
LGM: *To Maarssen?*
User: (?)
DM: `the_route.trajectory.destination ?+ mas`
LGM: *I understood that you want to travel to Maarssen; is this correct?*
User: *Yes.*

The aim of the formulation of the second verification question is to clarify the intentions of the system and to encourage the user to provide a simple *Yes* or *No* answer. Here, the variation in formulation has a clear function in the dialogue: if the

system had used the same wording twice, the user would probably have reacted in the same way, and the recognition problem would have persisted.

Information status in dialogues

In 'monologue mode', the context model of the LGM records which information has been expressed by the system. When referring to this information, the system may produce a reduced anaphoric expression, e.g., a deaccented pronoun. The underlying assumption is that information in the context model does not require emphasis in expression, because it *given*, i.e., mutually known to user and system. However, this assumption cannot be maintained in a spoken dialogue system. Here, due to imperfect speech recognition, the system is often not certain of what is in the linguistic context. This means that the simple division between new and given information is insufficient: some intermediate information status is required. Below, we discuss how this influences the generation of referring expressions and accentuation.

Grounding

In a dialogue, information that has been put forward by one speaker cannot be regarded as immediately 'known' by the other. Therefore, in human dialogues, the speakers continually react to each other's utterances, signaling whether or not they understood the other. This is called *grounding* (Clark & Schaefer 1989; Traum 1994): the interactive process of adding information to the speakers' mutual knowledge or common ground. Grounding typically proceeds in two phases: a presentation phase in which one speaker sends some message to his conversation partner, and an acceptance phase in which the other signals whether the message came across correctly or not. Information introduced by one participant is only added to the common ground if it has been accepted by the other, either by giving explicit positive feedback such as saying *OK* or by simply continuing the conversation.

In OVIS, information that is contributed by the user is never immediately grounded, i.e., regarded as known by the system. Instead, the system first asks a verification question to make sure that there has been no speech recognition error. The following dialogue fragment illustrates the OVIS grounding process.

(7)

a User: *I'd like to go to Maastricht.*
b LGM: *To Maarssen?*
c User: *No, to Maastricht.*
d LGM: *To Maastricht?*
e User: *Yes, that's correct.*
f LGM: *From where do you want to leave?*

In (7)a, the user presents the information that he or she wishes to travel to Maastricht. In (7)b, the system asks a verification question to check if it has recognized the user's utterance correctly, thereby (*i*) signaling to the user that the information presented in (7)a is not yet grounded (acceptance phase) and (*ii*) presenting what the system assumes the user has said (starting a subordinate presentation phase). In (7)c the user signals that the system's assumption is incorrect (non-acceptance of (7)b) and presents the correct information once again. Again, the system asks a verification question to check if there has been a speech recognition error. This time, the user reacts with a confirmation, signaling that the destination has been recognized correctly. (Note that (7)b-e are part of the overall acceptance phase for (7)a, but are also involved in subordinate presentation and acceptance cycles.) Finally, in (7)f) the system implicitly shows its acceptance by asking a new question. All in all, it has taken user and system five dialogue turns (from (7)b to (7)f) to ground the information provided by the user in (7)a.

In OVIS, information that has been presented by the user, but not yet grounded by the system, is assigned a special information status: that of *pending* information[7] (cf. the distinction between grounded and ungrounded information by Poesio & Traum (1997) and Matheson *et al.* (2000)). Clearly, this information should be represented in the LGM's context model, but at the same time it cannot be regarded as *given* in the sense of being mutually known. In the next sections it is discussed how this new, mixed information status relates to the generation of referring expressions and accentuation.

Referring expressions

In 'monologue mode', the LGM usually refers to items in the context model using reduced, anaphoric expressions.[8] In 'dialogue mode', however, the context model contains both information that is given (grounded) and information that is still pending (not grounded yet). This raises the question whether pending information can be anaphorically referred to as well. As argued by Dekker (1997), Groenendijk *et al.* (1997) and Poesio (1998), this is indeed possible, but only in contexts that do not require the speaker to have identified the referent (or even to have committed to its existence). Examples are questions and a modal statements, such as B(c-d) in example (8) from Poesio (1998). In assertions, on the other hand, anaphoric references to pending information are only marginally acceptable, as shown

[7] Material introduced by the system is assumed to be immediately grounded by the user. This simplification reflects the fact that in general, the system has far more problems understanding the user than vice versa.

[8] Given that certain restrictions on the distance between anaphor and antecedent, and the presence of potential distractors are met (Krahmer & Theune 2002).

by B(a). Before making an assertion, B must first explicitly signal having grounded (and thus identified the referent of) A's utterance. This is done in B(b).

(8)

A: *There is an engine at Avon.*
B(a): ?? *It is red.*
B(b): *Right / Yeah / I see it. It is red.*
B(c): *Is it in working conditions?*
B(d): *It might be used to carry the boxcar.*

Based on Allwood (1995) and Clark (1996), Larsson (2003) classifies grounding behaviour in terms of the *action level* involved: contact, perception, understanding, or integration. Example (8) deals with grounding at the level of understanding (which includes referent identification): all of B's reactions indicate that B has correctly perceived A's utterance, but understanding is only indicated by the explicit acknowledgement in B(b). An example involving a problem at the perception level is (9) below. Here, B cannot use a pronoun or other anaphoric expression, until A has accepted the repeated material presented by B (thus solving B's perception problem).

(9)

A: *There is an engine at Avon.*
B: *An engine?*
A: *Yes.*
B: (Any of B(b-d) from (8).)

In OVIS, most references to pending information are like (9) in that they involve the perception level rather than the understanding level. As illustrated by (9), in such a situation the use of anaphoric expressions is impossible until the perceived information has been confirmed or corrected. Therefore, the system always uses a full expression when referring to pending information, even though the information is represented in the context model.

Usually, after the user has confirmed the pending information, grounding by the system immediately follows. There are cases, however, where this does not happen, for example when the user wants to travel to a town that does not have a train station, and therefore cannot be accepted as the value for the destination slot (see the last example in (1)). Here, in terms of Larsson's action levels, the information provided by the user has been perceived and understood, but it cannot be integrated by the system and is therefore not grounded. In such cases, anaphoric expressions may be used without any problem. This is illustrated in (10) below.

(10)

User: *I'd like to go to Uden.*
LGM: *To Uden?*
User: *Yes, that's correct.*
LGM: *Uden / that town does not have a train station.*

Summarizing, cross-speaker anaphoric references are sometimes permitted for ungrounded information, depending on the action level involved. If there is a problem at the understanding level (identification) that precludes grounding, anaphoric references are only allowed in the context of a question or a modal statement. If the problem is at the level of integration, anaphoric references are always allowed. Finally, in the case of (potential) problems at the perception level, reduced references are impossible; only verbatim repetitions can be used. In spoken dialogue systems, most grounding problems are at the perception level, and this strongly limits the use of cross-speaker anaphora.

Accentuation

Contextually given words are generally deaccented, except in case of contrast (see Theune (2002)). For instance, in (8) all occurrences of the pronoun *it* would normally be pronounced without a pitch accent, and if speaker B had said *There is another engine at Bath*, the word *engine* would probably have been deaccented. Still, human speakers sometimes do accent contextually given, non-contrastive words in order to signal a problem at either the perception or the understanding level. Empirical evidence for this is given by Swerts *et al.* (1998), who studied the prosody of repetitive utterances ('repeats') in Japanese. They found that repeats signaling grounding, as in (11), are prosodically distinct from those signaling a possible communication error, as in (12). The latter generally have a relatively high pitch, as well as other marked prosodic features.

(11)

A: and then you transfer to the Keage line ...
B: Keage line
A: which will bring you to Kyoto station

(12)

A: and that is the Keage line ...
B: KEAGE line?
A: that's right, Keage line

Grice & Savino (1997) studied a corpus of map task dialogues between speakers of Bari Italian. In their corpus, accentuation of repeated material occurs if the instruction follower has no confidence in the instructions, for instance because the instructor refers to a landmark that is not on the map. Here, accentuation indicates a problem with understanding (identification) rather than perception.

In short, speakers tend to accent contextually given information to signal understanding or perception problems. Conversely, deaccentuation usually signals acceptance at these levels. For the OVIS system, this means that deaccenting repeated information in a verification question is unwanted, as

it may create the impression that the information is grounded by the system and can no longer be corrected.[9] Therefore, the simple strategy of deaccenting previously mentioned information has been adapted to make an exception for information that is being verified.

In cases like (10), where the system cannot integrate (and thus ground) the pending information, there seem to be two possibilities. Either the system can use a full, accented referring expression (UDEN) to convey a lack of confidence or it can use a deaccented, reduced description (*that town*) to indicate that the referent was successfully identified. Currently, the system always chooses the first option, to indicate that the user has provided an invalid value of the destination slot. As a general rule, in OVIS only grounded information is deaccented, even though this restriction may be too strong in some cases. For examples like (10), it should be empirically tested which is the best accentuation strategy.

All in all, for accentuation we see a somewhat similar picture as for referring expressions. Both deaccentuation and the use of reduced anaphoric descriptions are permissible across speakers in a dialogue. However, when there are potential problems at the perception level, human speakers generally do not use deaccented or reduced descriptions when referring to the other's utterance. In the OVIS system, where misrecognitions are not uncommon, the same strategy is adopted in order to achieve a natural dialogue, which runs as smoothly as possible.

Conclusions and future work

This paper has discussed the generation of system utterances in a spoken dialogue system called OVIS. The restriction that in a dialogue, the system's utterances should fit within one coherent dialogue turn gave rise to several modifications of the LGM. Due to the strict division between dialogue management and language generation, the linguistic and practical constraints of the LGM cannot be anticipated by the dialogue manager, so that sometimes the LGM can carry out only part of its instructions. This is somewhat similar to what happens in human sentence production, where there may be a mismatch between the speaker's conceptual intentions and the possibilities of formulation (Kempen (1977):260). The occasional occurrence of such mismatches in OVIS makes it important to send feedback from the LGM to the dialogue manager, which needs to know exactly which message has been conveyed to the user, to be able to react correctly to the user's subsequent utterance. This required feedback seems a reasonable price to pay for a strictly modular system architecture.

To achieve a smooth and natural dialogue, producing context-sensitive system prompts is very important. Therefore, a brief overview has been given of the influence of dialogue context on information status, and its consequences for the generation of referring expressions and accentuation. One of the observations that have been put forward is that in dialogues, an intermediate information status is required for information that is in the linguistic context but has not yet been grounded (i.e., is not yet regarded as mutually known). Grounding is only possible when the information provided by the user has been perceived correctly and can be integrated by the system. In a spoken dialogue system, perception is often hampered by imperfect speech recognition. Until the absence of recognition errors has been established, the system should not use a deaccented or reduced description to refer to information that was provided by the user.

One form of context-sensitivity which is currently not addressed in the dialogue version of the LGM, is adapting the wording of the system utterances to that of the user. In OVIS, the LGM knows which information has (presumably) been provided by the user, but not the words that were used to do so. As a consequence, the system sometimes uses other words than the user when referring to the same thing, for instance when the user wants to travel *next Friday* and the system subsequently refers to that date as *March the 28th*. Such discrepancies may create the impression that the system is correcting the user, and in the worst case cause considerable confusion. To be able to improve on this, the LGM should have access to the syntactic analysis of the user's utterances, and the generation algorithm should be adapted so that this information can be actually used to influence the wording of the generated prompts. This is left as future work

Finally, it should be noted that prompt design for OVIS was (partially) based on the results of user tests with two comparable, commercial Dutch train information systems (Weegels 2000). The OVIS system itself, including the language generation component as described in this paper, has not been formally evaluated yet.

Acknowledgements

Thanks are due to the two anonymous reviewers who provided useful comments on an earlier version of this paper, which is based on Chapter 5 from Theune (2000). Thanks also to Staffan Larsson for providing me with a draft version of his contribution to this workshop.

[9] This is similar to the effect of the 'implicit' verification questions asked in an earlier version of the OVIS system, which verified one slot value while asking for the value of another (e.g., *At what time do you want to travel to Maarssen?*). Users often did not realize that the information mentioned in such questions could still be corrected.

References

Allwood, J. 1995. An activity based approach to pragmatics. Gothenburg Papers in Theoretical Linguistics 76, University of Göteborg.

Bod, R. 1998. Spoken dialogue interpretation with the DOP model. In *Proceedings of COLING-ACL'98*, 138–144.

Clark, H., and Schaefer, E. 1989. Contributing to discourse. *Cognitive Science* 13:259–294.

Clark, H. 1996. *Using Language*. Cambridge: Cambridge University Press.

Deemter, K. v., and Odijk, J. 1997. Context modeling and the generation of spoken discourse. *Speech Communication* 21(1/2):101–121.

Dekker, P. 1997. On first order information exchange. In Benz, A., and Jäger, G., eds., *Proceedings of Mundial 97*.

Grice, M., and Savino, M. 1997. Can pitch accent type convey information status in yes-no questions? In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 29–33.

Groenendijk, J.; Stokhof, M.; and Veltman, F. 1997. Coreference and modality in the context of multi-speaker discourse. In Kamp, H., and Partee, B., eds., *Context Dependence in the Analysis of Linguistic Meaning*, 195–216.

Kempen, G. 1977. Conceptualizing and formulating in sentence production. In Rosenberg, S., ed., *Sentence Production: Developments in Research and Theory*. Hillsdale: Lawrence Erlbaum Associates. 259–274.

Klabbers, E. 2000. *Segmental and Prosodic Improvements to Speech Generation*. Ph.D. Dissertation, Eindhoven University of Technology.

Krahmer, E., and Theune, M. 2002. Efficient context-sensitive generation of referring expressions. In van Deemter, K., and Kibble, R., eds., *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*. Hillsdale: CSLI Publications. 223–264.

Larsson, S. 2003. Generating feedback and sequencing moves in a dialogue system. In *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*. These proceedings.

Matheson, C.; Poesio, M.; and Traum, D. 2000. Modelling grounding and discourse obligations using update rules. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics*.

Noord, G. v.; Bouma, G.; Koeling, R.; and Nederhof, M. 1999. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering* 5(1):45–93.

Poesio, M., and Traum, D. R. 1997. Conversational actions and discourse situations. *Computational Intelligence* 13(3).

Poesio, M. 1998. Cross-speaker anaphora and dialogue acts. In *Proceedings of the Workshop on Mutual Knowledge, Common Ground and Public Information, ESSLLI'98*.

Strik, H.; Russel, A.; van den Heuvel, H.; Cucchiarini, C.; and Boves, L. 1997. A spoken dialog system for the dutch public transport information service. *International Journal of Speech Technology* 2:119–129.

Swerts, M.; Koiso, H.; Shimojima, A.; and Katagiri, Y. 1998. On different functions of repetitive utterances. In *Proceedings of ICSLP'98*, volume 2, 483–486.

Theune, M.; Klabbers, E.; de Pijper, J.; Krahmer, E.; and Odijk, J. 2001. From data to speech: A general approach. *Natural Language Engineering* 7(1):47–86.

Theune, M.; Klabbers, E.; Odijk, J.; and de Pijper, J. 1997. Computing prosodic properties in a data-to-speech system. In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 39–45.

Theune, M. 2000. *From Data to Speech: Language Generation in Context*. Ph.D. Dissertation, Eindhoven University of Technology.

Theune, M. 2002. Contrast in concept-to-speech generation. *Computer Speech and Language* 16(3/4):491–531.

Traum, D. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. Dissertation, University of Rochester.

Veldhuijzen van Zanten, G. 1998. Adaptive mixed-initiative dialogue management. In *Proceedings of IVTTA 1998*, 65–70.

Weegels, M. 2000. Users' conceptions of voice-operated information services. *International Journal of Speech Technology* 3(2):75–82.